الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد (م.و.ت.م.ب)

COURS DE SYSTÈME D'EXPLOITATION

SÉRIE 04

OBJECTIF PÉDAGOGIQUE :

À l'issue de cette série le stagiaire doit être capable de **c**onnaître :

1- Les mécanismes de gestion de la mémoire.
2- Les différentes stratégies d'allocation de la mémoire.

PLAN DE LA LEÇON :

INTRODUCTION :

La mémoire physique sur un système se divise en deux catégories :

- La mémoire vive: composée de circuit intégrés, donc très rapide.
- La mémoire de masse (secondaire): composée de supports magnétiques (disque dur, bandes magnétiques...), qui est beaucoup plus lente que la mémoire vive.

La mémoire est une ressource rare. Il n'en existe qu'un seul exemplaire et tous les processus actifs doivent la partager. Si les mécanismes de gestion de ce partage sont peu efficaces l'ordinateur sera peu performant, quelque soit la puissance de son processeur. Celui-ci sera souvent interrompu en attente d'un échange qu'il aura réclamé. Si un programme viole les règles de fonctionnement de la mémoire, il en sera de même.

I- OBJECTIFS :

La gestion de la mémoire est un difficile compromis entre les performances (temps d'accès) et la quantité (espace disponible). On désire en effet tout le temps avoir le maximum de mémoire disponible, mais l'on souhaite rarement que cela se fasse au détriment des performances.

La gestion de la mémoire doit de plus remplir les objectifs suivants :

- Permettre le partage de la mémoire (pour un système multitâches) ;
- Permettre d'allouer des blocs de mémoire aux différentes tâches ;
- Protéger les espaces mémoire utilisés (empêcher par exemple à un utilisateur de modifier une tâche exécutée par un autre utilisateur) ;
- Optimiser la quantité de mémoire disponible, notamment par des mécanismes d'extension de la mémoire.

II- FONCTIONS D'UN GESTIONNAIRE DE MÉMOIRE :

Le gestionnaire de mémoire est un sous-ensemble du système d'exploitation. Son rôle est de partager la mémoire entre l'OS et les diverses applications. Le terme "mémoire" fait surtout référence à la mémoire principale, c'est à dire à la RAM, mais la gestion de celle-ci demande la contribution de la mémoire auxiliaire (mémoire de masse, spacieuse mais lente) et à la mémoire cache (rapide mais de taille restreinte).

Voici quatre fonctions qu'on attend du gestionnaire de mémoire :

- **L'allocation de la mémoire aux processus :**

  - Répertorier les emplacements libres de la mémoire
  - Allouer la mémoire nécessaire aux nouveaux processus
  - Récupérer la mémoire des processus qui s'achèvent.

Cette récupération peut nécessiter une réallocation des processus en cours pour optimiser l'emploi de la mémoire. La zone mémoire attribuée à un processus peut donc changer au cours de son exécution.

- **La protection**

Il faut s'assurer que les adresses générées par chaque processus ne concernent que la zone mémoire qui lui est impartie, sans quoi, l'intégrité du système d'exploitation et des autres processus n'est pas garantie.

Certaines zones mémoire doivent pourtant servir simultanément à plusieurs processus : le code de fonctions servant à plusieurs applications qui tournent en parallèle ou les données utilisées simultanément par divers processus

- **La segmentation de l'espace d'adressage :**

Les programmes sont subdivisés en segments : le code, les données modifiables, celles qui ne le sont pas, la pile. On attend donc du gestionnaire de mémoire qu'il permette la segmentation de l'espace d'adressage des programmes pour les raisons suivantes :

- Pouvoir coder les segments séparément et les paramétrer en fonction de l'application.
- Permettre des degrés de protection différents selon les segments (lecture seule, exécution...)
- Accepter le partage de certains segments.

- **La mémoire virtuelle**

Elle offre aux applications une mémoire de taille supérieure à celle de la mémoire principale.

L'espace d'adressage qu'offre la mémoire centrale est parfois insuffisant. Les disques suppléent à cette insuffisance en fournissant une mémoire auxiliaire plus vaste mais plus lente et qui n'est pas directement accessible au processeur.

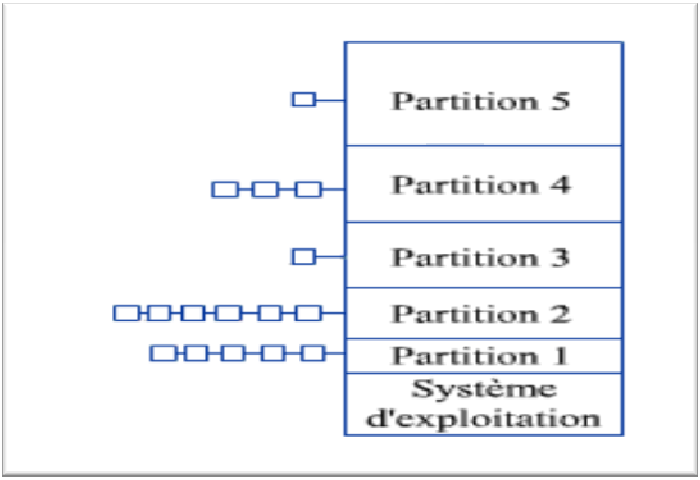
II- STRATÉGIE D'ALLOCATIONS DE LA MÉMOIRE :

Plusieurs processus doivent se partager la mémoire sans empiéter sur l'espace réservé au système d'exploitation ni aux autres processus. Quand un processus se termine, le S.E. doit libérer l'espace mémoire qui lui était alloué pour pouvoir y placer de nouveaux processus.

1-Allocation en zones contiguës :

1.1- Partitions fixes :

Le plus simple est de diviser la mémoire en partitions fixes dès le démarrage du système. Les partitions sont de différentes tailles pour éviter que de grandes partitions ne soient occupées que par de petits processus. Le gestionnaire de mémoire, en fonction de la taille des processus, décide quelle partition lui allouer pour ne pas gaspiller trop de mémoire.



Une file d'attente est associée à chaque partition. Quand vient une nouvelle tâche, le gestionnaire détermine quelle est la plus petite partition qui peut la contenir puis place cette tâche dans la file correspondante.

Le fait d'éviter d'allouer une partition trop grande à un petit processus conduit parfois à des aberrations. Il arrive que des partitions plus grandes restent inutilisées alors que se forment ailleurs des files interminables de petits processus. La mémoire est donc mal utilisée.

Une autre solution est de créer une file unique. Lorsqu'une partition se libère, on consulte la file pour trouver la tâche qui l'occuperait de manière optimale.

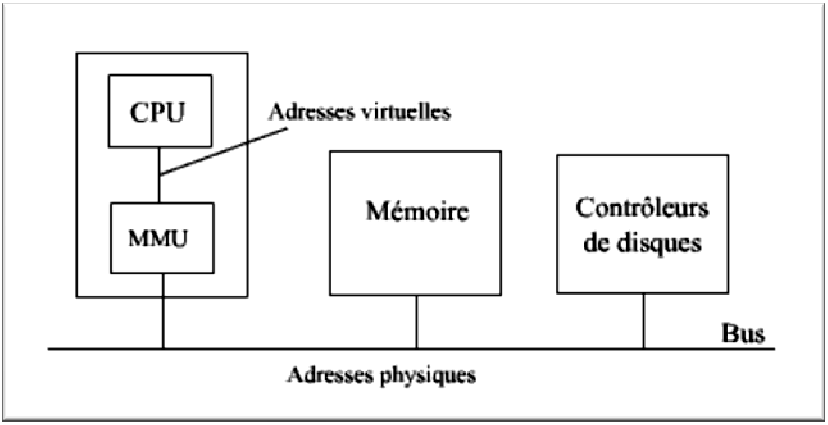
Le risque est que les petites tâches soient pénalisées. Une parade est de conserver une petite partition au moins qui ne sera accessible qu'aux petites tâches. Une autre solution, serait de dire qu'un processus ne peut être ignoré qu'au maximum un certain nombre de fois. Après **n** refus, il prendra place dans une partition même si la partition est bien plus grande qu'il ne faut.

1.2- Partitions variables :

Une autre manière d'éviter les emplacements mémoires inoccupés en fin de partitions est d'allouer aux processus des espaces qui correspondent exactement à l'espace qui leur est utile.

Au fur et à mesure que les processus se créent et se terminent, des partitions s'allouent et se libèrent laissant des zones mémoires morcelées et inutilisables.

La mémoire se fragmente et est de plus en plus mal employée. Il faudrait la compacter en déplaçant régulièrement les processus mais cette tâche supplémentaire ralentit le système.



1.3- Conclusion :

Le partitionnement de la mémoire que ce soit avec des partitions de tailles fixes ou de tailles variables, ne permet pas d'utiliser la mémoire au mieux.

2- La pagination :

Les processus requièrent des espaces d'adresses continus. On a vu que cela est difficilement réalisable en découpant la mémoire en partions dont les tailles correspondent à celles des processus. La pagination est une technique d'allocation de la mémoire bien plus efficace. Elle fournit aux processus des *espaces d'adresses* séquentiels à partir d'*espaces mémoire* discontinus.

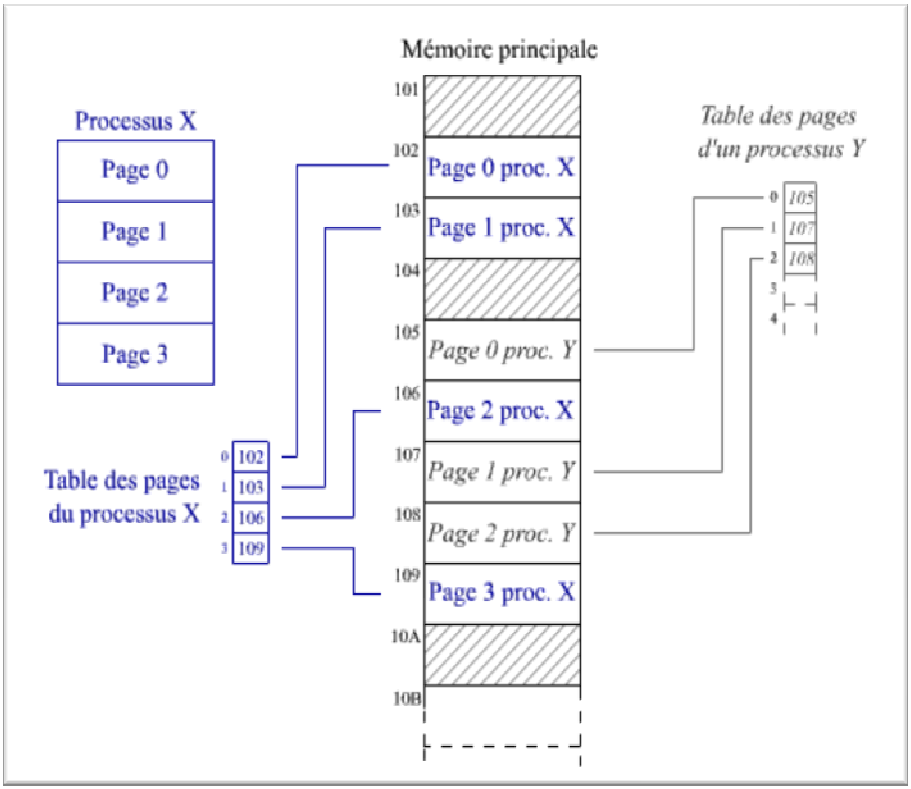
La pagination consiste à diviser la mémoire et les processus en blocs de mêmes tailles appelés **pages**. Les *pages mémoire* sont souvent appelées *"frames"* ou *"cadres"* tandis que les *pages de processus* sont simplement appelées *"pages"*.

Les pages (de processus) ne sont pas toutes simultanément actives ; elles ne sont donc pas nécessairement toutes présentes simultanément dans la mémoire principale. Les pages inactives attendent sur le

disque. L'*espace d'adressage* est donc *virtuel* sa taille peut être supérieure à celle de la mémoire réelle.

Les processeurs disposent actuellement d'un dispositif, le MMU *"Memory Manager Unit"* qui permet de placer des processus en mémoire sans nécessairement placer les pages de processus dans des cadres de pages contigus. On distingue les adresses logiques qui se réfèrent aux pages de processus des adresses physiques qui se réfèrent aux cadres de pages.

Voyons à présent comment l'unité de gestion mémoire (MMU) met en correspondance les adresses physiques et logiques. Elle contient pour ce faire une table de pages où sont inscrits les numéros des cadres de pages.



➢ Fonctionnement des tables de pages :

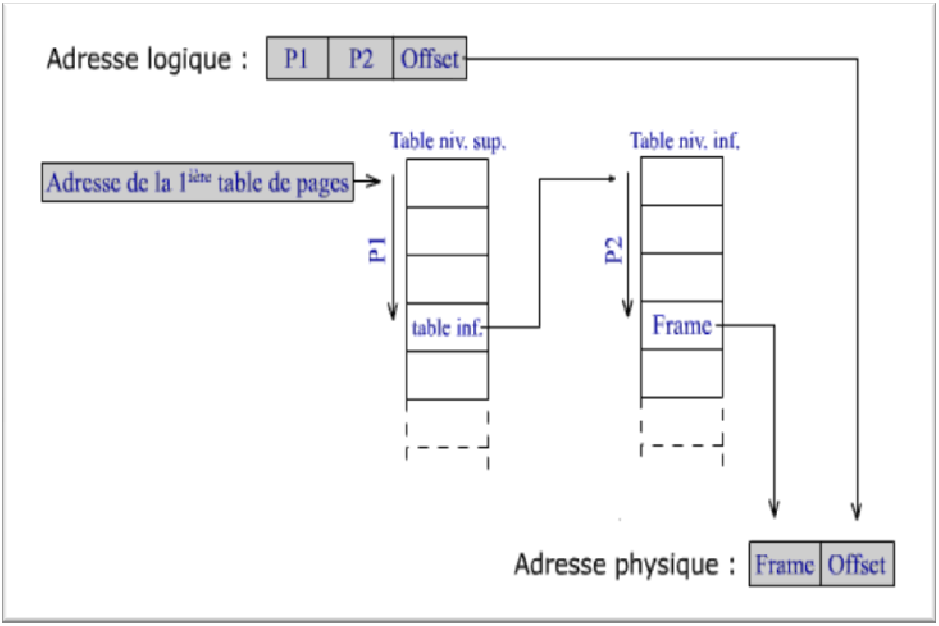
L'adressage se fait au moyen de *numéros de pages* et d'*offsets*. L'offset (= déplacement ou décalage) est la position relative au début de la page.

L'adresse logique est composée du numéro de *page de processus* et d'un offset.

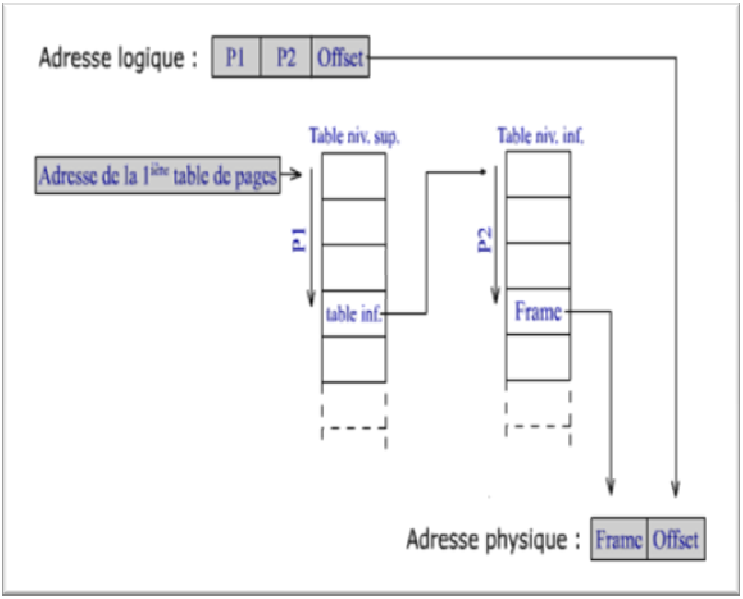
L'adresse physique correspondante est formée à partir du numéro du *cadre de page* où est chargé la page de processus et du même offset que celui de l'adresse logique.

Le numéro du cadre de page est consigné dans une table des pages associée au processus. On y retrouve le numéro du cadre de page en se servant du numéro de page de processus comme d'un index.

Pagination simple

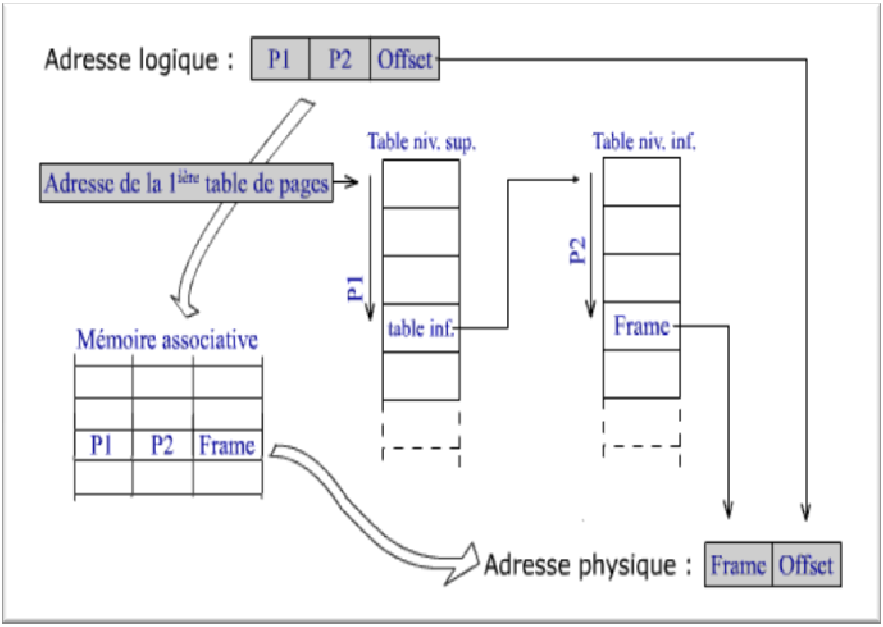


Le nombre de pages étant souvent très grand les tables des pages deviennent volumineuses et peuvent même occuper ... plusieurs pages. On les fractionne donc en plusieurs niveaux : une table de page de niveau supérieur dont chaque élément pointe vers une table de niveau inférieur. L'adresse logique contient dès lors deux nombres pour aboutir au numéro de page. Le premier sert d'index dans la table de niveau supérieur, le second sert d'index dans la table du niveau suivant.



Tables de pages multi niveaux

Ces accès multiples à différentes pages pour aboutir à l'adresse finale ralentissent fortement l'adressage. On évite de répéter ces recherches en notant les correspondances trouvées entre les adresses logiques et les adresses physiques dans une mémoire associative. Ce qui permet ensuite de retrouver presque immédiatement les correspondances les plus récentes.



NB. L'espace d'adressage est perçu par le programmeur comme une suite continue d'octets. La subdivision de l'adresse en numéros de page et d'offset est transparente. Elle est prise en charge par le matériel.

3- La segmentation :

Chaque processus est constitué d'un ensemble de segments. Chaque segment est un espace linéaire.

Les segments sont des espaces d'adressages indépendants de différentes longueurs et qui peuvent même varier en cours d'utilisation. Ils correspondent à des subdivisions logiques déterminées par le programmeur ou par le compilateur.

Les segments contiennent des informations de même nature : le code, les données, la pile, des tables, etc. Il est dès lors possible d'attribuer des protections adaptées à chaque type de segment : un segment de code peut être déclaré en exécution seule, une table de constantes en lecture seule mais pas en écriture ni en exécution. Certaines zones de code en exécution seule peuvent être partagées par plusieurs

processus ; cela se fait par exemple pour des bibliothèques de sous-programmes.

L'accès aux segments se fait via une table de segments.

Chaque entrée de la table comporte l'adresse de départ du segment et sa taille.

L'adresse logique est constituée du numéro de segment et d'un offset. Contrairement aux pages dont le fonctionnement est transparent pour le programmeur, les segments sont des entités logiques qu'il connaît et manipule. Il distingue les deux informations contenues dans l'adresse : le numéro du segment et l'offset.

Le numéro de segment sert d'index pour retrouver l'adresse du début du segment dans la table de segment. Cet offset doit être inférieur à la taille du segment consignée elle aussi dans la table de segment. Si ce n'est pas le cas, une erreur est générée qui provoque l'abandon du programme. L'offset est ensuite ajouté à l'adresse de début de segment pour former l'adresse physique.

4- Segmentation avec pagination :

La segmentation et la pagination concernent des problèmes différents. Ce sont deux techniques qui peuvent se combiner :

- La segmentation découpe les processus en zones linaires pouvant être gérées différemment selon que ces segments sont propres au processus, qu'ils sont partagées, lus, écrits ou exécutées et de manière à protéger les processus entre eux.

- La pagination découpe la mémoire en pages non contiguës mais de même taille. Elle procure aux processus des espaces d'adresse continus (nécessaires aux segments). Les pages mémoires peuvent n'être allouées que lorsqu'un processus en a besoin. On obtient de la sorte une mémoire virtuelle de taille supérieure à la mémoire réelle.

Les systèmes d'exploitation qui gèrent ces deux techniques simultanément administrent **une table de segments et plusieurs tables de pages**.

Un segment peut contenir plusieurs pages mais toutes ne doivent pas nécessairement être présentes en mémoire à tout moment. On ne garde en mémoire que celles qui sont réellement utilisées. Chaque segment a sa propre table de pages.

I.V- NOTION DE MÉMOIRE VIRTUELLE :

La taille d'un processus doit pouvoir dépasser la taille de la mémoire physique disponible, même si l'on enlève tous les autres processus. Afin d'assurer une extension de la mémoire il existe 2 manières **:**

- En découpant un programme en une partie résidente en mémoire vive et une partie chargée uniquement en mémoire lorsque l'accès à ces données est nécessaire.

- En utilisant un mécanisme de mémoire virtuelle, consistant à utiliser le disque dur comme mémoire principale et à stocker uniquement dans la RAM les instructions et les données utilisées par le processeur. Le système d'exploitation réalise cette opération en créant un fichier temporaire (appelé fichier SWAP, traduit "fichier d'échange ") dans lequel sont stockées les informations lorsque la quantité de mémoire vive n'est plus suffisante. Cette opération se traduit par une baisse considérable des performances, étant donné que le temps d'accès du disque dur est extrêmement plus faible que celui de la RAM. Lors de l'utilisation de la mémoire virtuelle, il est courant de constater que la LED du disque dur reste quasiment constamment allumée et dans le cas du système Microsoft Windows qu'un fichier appelé "win386.swp" d'une taille conséquente, proportionnelle aux besoins en mémoire vive, fait son apparition.

La mémoire virtuelle est une mémoire idéale, dont les adresses commencent à 0, et de capacité non limitée ; elle a pour but principal de pouvoir exécuter des processus sans qu'ils soient logés en mémoire en leur totalité ; sans recours à la mémoire virtuelle, un processus est entièrement chargé à des adresses contiguës ; avec le recours à la mémoire virtuelle, un processus peut être chargé dans des pages ou des segments non contigus. On appelle Espace Virtuel l'ensemble des adresses possibles ; il est fixé par le nombre de bits qui constitue une adresse virtuelle. On parlera d'adresse réelle et d'adresse virtuelle.

Toute la difficulté consiste à traduire une adresse virtuelle (A.V.) en une adresse réelle (A.R.) accessible sur la machine.

EXERCICES D'APPLICATIONS:

EXERCICE N°01 : Segmentation

On considère la table des segments suivante pour un processus P1 :

Segment	Base	Limite
0	540	234
1	1254	128
2	54	328
3	2048	1024
4	976	200

1- Calculez les adresses réelles correspondant aux adresses virtuelles suivantes (vous signalerez éventuellement les erreurs d'adressage) :(0:128), (1:100), (2:465), (3:888), (4:100), (4:344).

2- L'adresse virtuelle (4,200) est-elle valide ?

Rappel : Les adresses sont données sous la forme (n°segment: déplacement)

EXERCICE N°02 : Pagination

Dans un système paginé, les pages font 256 mots mémoire et on autorise chaque processus à utiliser au plus 4 cadres de la mémoire centrale. On considère la table des pages suivante du processus P1 :

Page	0	1	2	3	4	5	6	7
Cadre	011	001	000	010	100	111	101	110
Présence	oui	non	oui	non	non	non	oui	non

1- Quelle est la taille de l'espace d'adressage du processus P1 ?
2- De combien de mémoire vive dispose ce système ?
3- Calculez les adresses réelles correspondant aux adresses virtuelles suivantes (vous signalerez éventuellement les erreurs d'adressage) :240, 546, 1578, 2072
4- Que se passe-t-il si P1 génère l'adresse virtuelle 770 ?
5- On considère l'adresse virtuelle suivante: 0000 0000 0000 0111. Sachant que les 4 bits de poids fort désigne le numéro de page et que 12 bits suivants représentent le déplacement dans la page, donnez l'adresse physique (exprimée en binaire) correspondant à cette adresse.

EXERCICE N°03 : Citez les objectifs de la gestion de la mémoire

CORRECTION DES EXERCICES :

EXERCICE N°01 :

1- L'adresse physique s'obtient en ajoutant l'adresse de base du segment au déplacement dans le segment, mais à condition que le déplacement ne soit pas supérieur à la taille du segment moins 1 (on compte le déplacement en partant de 0) :

- (0:128) : déplacement valide (128<234). Adr_physique = base + limite =540+128 = 668.
- (1:100) : déplacement valide (100<128). Adr_physique = bas +limite= 1254+100 = 1354.
- (2:465) : déplacement invalide (465>328).
- (3:888) : déplacement valide (888<1024). Adr_physique= base+limite= 2048+888= 2936.
- (4:100) : déplacement valide (100<200). Adr_physique = base+ limite= 976+100 = 1076.
- (4:344) : déplacement invalide car (344>200)

2- Non. Dans un segment de longueur 200, les déplacements valides sont dans l'intervalle [0-199]

EXERCICE N°02 :

1- L'espace d'adressage du processus est l'espace d'adressage virtuel formé par les pages. Comme il y a 8 pages, la taille de l'espace virtuel est de 8*256 = 2048 mots.

2- Comme les cadres sont numérotés sur 3 bits, il y a 23= 8 cadres. Taille d'un cadre = taille d'une page donc la mémoire physique comporte 8*256 =2048 mots (= 2Ko).

3- La conversion d'une adresse virtuelle en adresse réelle est réalisée de la façon suivante **:**

a. Calcul du numéro de la page et du déplacement dans la page.

b. Recherche dans la table de pages de l'entrée qui correspond à la page de façon à en déduire le numéro du cadre.

c. L'adresse physique (réelle) est obtenue en ajoutant le déplacement à l'adresse physique de début du cadre.

Voici le détail des calculs pour les adresses demandées :

- 240 = 0*256 + 240 → page = 0 et déplacement = 240

D'après la table des pages, cadre = 3. D'où Adr_phys =3*256 + 240 = 1008

- 546 = 2 *256 + 34 → page = 0 et déplacement = 34.

D'après la table des pages, cadre = 0. D'où Adr_phys =0*256+ 34 = 34.

- 1578 = 6*256 + 42 → page = 6 et déplacement = 42.

D'après la table des pages, cadre = 5. D'où Adr_phys =5*256 + 42 = 1322.

- 2072 est en dehors de l'espace d'adressage virtuel du processus (2048 mots).

4- 4**)** 770 = 3*256 + 2. Il s'agit d'une adresse située dans la page 3. Or d'après la table des pages, cette page n'est pas présente en mémoire. Une référence à cette adresse provoquera donc un défaut de page.

5- D'après la table de pages, cette page se trouve dans le cadre 010. L'adresse physique s'obtient donc simplement en substituant aux 4 bits de poids fort de l'adresse virtuelle les 3 bits du numéro de cadre : 010 0000 0000 0111**.**

EXERCICE N°03 :

- Protection : par défaut, un processus ne doit pas pouvoir accéder à l'espace d'adressage d'un autre
- Partage : s'ils le demandent, plusieurs processus peuvent partager une zone mémoire commune,
- Réallocation: les adresses vues par le processus (adresses relatives ou virtuelles) sont différentes des adresses d'implantation (adresses absolues).

RÉFÉRENCES BIBLIOGRAPHIQUES :

1. CT BULABULA DEMA Faustin, Cours de Systèmes d'exploitation et le Bureautique I, ISP/Bukavu en G1 IG, Inédit, 2004-2005
2. Ass TASHO KASONGO, Cours de l'Introduction à l'Informatique, ISP/Bukavu en G1 IG, Inédit, 2006-2007
3. Ass KYENDA SULIKA Dieu-donné, Cours de Systèmes d'exploitation, ISP/Bukavu en G1 IG, Inédit, 2000-2001.
4. AUMIAUX, M. Initiation à l'informatique de gestion, 2ème éd. Masson, Paris, 1983.
5. CAPRON, H.L. Computer, a tool for information age, 3ème ed. Upper saddle river, New Jersey, 2000.
6. CHAUMEL J.L.: L'implantation d'une technologie, nouvelle leçon corrigée, ed. Hériat, Montréal, 1989.
7. DANIEL, C. : Organiser le développement de la micro-informatique, Ed. d'organisations, Paris, 1987.
8. DELEPONE, J.F., Introduction théorique à l'informatique, Africa Computing, Abidjan, 2004.
9. DONALD H. : L'informatique : Un instrument de la gestion, Mc Graw-Hill, 1980.
10. DULONG, A. et SUTTER, E. : Les technologies de l'information, Paris, 1990,
11. JAVEAU, C. : Enquête par questionnaire, éd. ULB, 1995.
12. LOOIJEN M. : Information Systems : Management, Control and Maintenance, Kluwer Bedrijfsinformatique, 1998. .

WEBOGRAPHIE :

http://www.bestcours.com/systeme-exploitation

http://depinfo.u-cergy.fr/~pl/docs/slidesOS.pdf

http://deptinfo.cnam.fr/Enseignement/CycleA/AMSI/transparents_systemes/14_gestion_memoire.pdf